REMARKS

Claims 1-3 and 5-34 are pending. Claims 1, 7, 17, 21, and 31 are independent. Claims 1, 3, 11, and 25 are amended. Claims 1-3 and 5-34 are rejected under 35 U.S.C. § 103. Reconsideration of the rejection in view of the foregoing amendments and following remarks is respectfully requested.

Although unnecessary and without prejudice or waiver, claim 1 has been amended to indicate that the instance of a main process not only has response processing code with exception handling, but also error compensation performed synchronously for asynchronous messaging.

In summary, the Office Action admits "Khodabakchian is silent with reference to the instance further configured to process the response using the exception handling code within the instance." (Office Action, p. 4). Since, according to the claims, the exception handling code must include error compensation, Khodabakchian also fails to disclose error compensation within the instance. The Office Action turns to Broussard for a "do nothing" try catch block in an instance, but Broussard clearly teaches that there is no error handling or error compensation in the instance. Thus, the Office Action fails to address the limitation requiring error compensation in the instance.

Additionally, although unnecessary and without prejudice or waiver, a common amendment is made to dependent claims 3, 11, and 25. Support for the amendment may be found, for example, in paragraph 0060.

Telephone Conversation With Examiner

Examiner Anya is thanked for the telephone conversation conducted on May 12, 2010. Differences between asserted art and claimed subject matter were discussed. Examiner Anya agreed to take Applicant's representatives comment under consideration. No agreements were reached.

Rejection of Claims 1-3 and 5-34 under 35 U.S.C. § 103(a)

Claims 1-3 and 5-34 are rejected under 35 U.S.C. § 103(a) as being unpatentable over (1) U.S. Patent Application Publication No. 2003/0093500, by Khodabakchian *et al.* (hereinafter referred to as "Khodabakchian") in view of at least one of: (2) U.S. Patent No. 7,036,045, issued to Broussard (hereinafter referred to as "Broussard"); and (3) U.S. Patent Application Publication No. 2003/0204835, by Budhiraja *et al.* (hereinafter referred to as "Budhiraja"). Specifically, claims 1, 2, 5-10, 12-24, and 26-34 are rejected in view of Khodabakchian and Broussard and claims 3, 11, and 25 are rejected in view of Khodabakchian, Broussard and Budhiraja. (Office Action, pp. 2-10.) Applicants respectfully traverse the rejections.

Previous remarks remain applicable and are repeated below. In addition, Applicants point out that the rejection overlooked claimed subject matter. The claims state that asynchronous messaging be handled synchronously by an instance of a process having exception handling specifying error compensation within the instance:

> **instance** of [a] process . . . **including** . . . **exception handling code specifying error compensation**;
> . . . handle exceptions using **the exception handling code specifying error compensation within the instance** thereby handling asynchronous messaging errors synchronously within the instance. (Amended claim 1).

> **response processing code within the instance** . . . **has failure handling functionality specifying error compensation** thereby handling asynchronous messaging errors synchronously within the instance (Claims 7, 17, 21 and 31).

In contrast to the above claimed subject matter, Khodabakchian indicates that failures are not processed within an instance of a process, but instead are processed by an independent "exception handler" 202 on web service orchestration server 102.

Khodabakchian discusses a system "handling processes that interact with one or more asynchronous systems." Khodabakchian at ¶ 0005. Khodabakchian states that, "Web service orchestration server 102 contains multiple scenarios 110(1), 110(2) and 110(3)." Khodabakchian at ¶ 0023. "A scenario is a programming abstraction of a long-running process or a collaborative

process." Id. "An exception handler 202 processes exceptions that are generated by orchestration engine 200 when implementing the instructions and commands in a scenario 110. Exceptions may include, for example, a fault generated by a web service or a notice generated as a result of a web service timeout." Khodabakchian at ¶ 0025; Figure 2. Khodabakchian indicates that failures are not processed within an instance of a process, but instead are processed by an independent "exception handler" 202 on web service orchestration server 102. Id.

The Office Action's citation to paragraphs 0024 and 0025 of Khodabakchian does not teach or suggest the claim limitation. Instead, they merely state that "[a] visual scenario design tool allows users [to] define partners, containers, flows, compensation rules and exception handlers for a specific scenario." Khodabakchian, ¶ 0024. FIGS. 1 and 2 of Khodabakchian show a separate centralized exception handler 202 (FIG. 2) for all scenarios. Thus, there is not exception handling or error compensation in the scenarios.

Further, the Office Action admits "Khodabakchian is silent with reference to the instance further configured to process the response using the exception handling code within the instance." (Office Action, p. 4). Since, according to the claims, the exception handling code must include error compensation, Khodabakchian also fails to disclose error compensation within the instance.

The Office Action turns to Broussard for a "do nothing" try catch block in an instance, but even if the "do nothing" try catch block is in an instance, it clearly fails to teach or suggest error handling or error compensation in the instance. Thus, the Office Action fails to address the limitation requiring error compensation in the instance.

As previously pointed out, Khodabakchian discusses a system "handling processes that interact with one or more asynchronous systems." Khodabakchian at ¶ 0005. Khodabakchian states that, "Web service orchestration server 102 contains multiple scenarios 110(1), 110(2) and 110(3)." Khodabakchian at ¶ 0023. "A scenario is a programming abstraction of a long-running process or a collaborative process." Id. "An exception handler 202 processes exceptions that are

generated by orchestration engine 200 when implementing the instructions and commands in a scenario 110. Exceptions may include, for example, a fault generated by a web service or a notice generated as a result of a web service timeout." Khodabakchian at ¶ 0025; Figure 2. Khodabakchian indicates that failures are not processed within an instance of a process, but instead are processed by an independent "exception handler" 202 on web service orchestration server 102. Id.

The Office Action admits Khodabakchian fails to teach or suggest the claimed subject matter. (Office Action, p. 4). Khodabakchian is not merely "silent" on the subject of exception handling code in an instance. Khodabakchian clearly indicates that exception handling in a separate centralized process 202 (FIG. 2) that is not present in scenarios.

The Office Action alleges that Broussard teaches what Khodabakchian fails to teach. However, it is respectfully submitted, Broussard fails to teach what the Office Action says it does. Specifically, Broussard states as follows:

> Problems often occur, however, because many applications write code that catch exceptions that perform some type of default behavior. In Java, for example, exceptions are sometimes caught generically, such as with the statement:
>
> catch (Exception e) {/* do nothing */}.
>
> In this case, nothing is done except to consume the exception. While this may work fine for cases where the exceptions are intended, it may be insufficient to handle other, perhaps unintended, exceptions such as a ClassNotFoundException. In cases where the exception is due to some user error (e.g., configuration problem, missing classpath item, etc.), the program will behave badly, and there may be no visible sign of what the problem is. (Broussard, col. 1, ll. 30-43 (emphasis added)).
>
> Within application logic 50 is a piece of code containing a "try-catch" block 56 to deal with any "Exception e" errors that might occur while "Do some business logic" is executed. If an error occurs 58, an exception is created 60, and the verbose exceptions mode is checked 62. If verbose is enabled for the exception 64, then the exception is output along with its stack trace 66, and the application logic continues 52. (Broussard, col. 4, ll. 57-64).

Try {
Do some business logic
} catch (Exception e)
{do nothing} (FIG. 4 (block 56)).

As stated in Broussard's background section, "do nothing" means "nothing is done except to consume the exception." Thus, nothing is done in the application logic, 50, 52, let alone JVM 18. Broussard clearly states there is no error handling or compensation in the instance. Instead, steps 58-66 are only error logging, not error handling. There is no error compensation. FIG. 4 shows that error occurs 58 then exception created 60 then check mode 62 then if verbose enabled for exception 64 output exception/stack trace 66. Broussard makes quite clear that nothing is done upon catching the exception, i.e., "do nothing." Like Khodabakchian, any handling code specifying error compensation is elsewhere, not in the application logic 50.

Additionally, there is no legitimate motivation to modify Khodabakchian. Khodabakchian, like other references, handles errors in a separate process using centralized exception handler 202. The Office Action asserts that the motivation to change this so that Khodabakchian's scenarios handle exceptions is to "provid[e] a condition system with a mechanism for signaling and handling unusual conditions, including errors and warnings." This generic conclusion is clearly without merit because Khodabakchian already handles unusual conditions separately using Exception Handler 202.

The foregoing remarks apply equally well to all pending claims. Further, the Office Action admits that Khodabakchian fails to disclose storing the instance after a predetermined time as claimed in claims 3, 11 and 25, but alleges Budhiraja does. (Office Action, p. 9). It is respectfully submitted that this argument is also inaccurate.

The Office Action alleges that paragraphs 0037, 0041 and 0048 of Budhiraja teach, as claimed in claims 3, 11 and 25, "wherein storing the instance takes place after a predetermined time after the detection that the instance is waiting for the response." Specifically, the Office

Action alleges use of the word "checkpoint" teaches claims 3, 11 and 25. However, nowhere does Budhiraja teach what is claimed.

Checkpointing discloses nothing about setting a predetermined time to store an instance. Here is what Budhiraja actually says:

> [0037] Business process manager 408 has the ability to persistently save (checkpoint) and recover the state of the business process objects it creates. For the particular implementation shown, business process manager 408 uses a database (data store) for this purpose. Other implementations may use different persistent storage methods.
>
> [0041] The checkpointing ability of Business process manager 408 is extended so that name and label information is included with saved state information. This means that each time that business process manager 408 saves the state of a business process object, it also saves the name and label associated with the business process model used to create that business process object.
>
> [0048] Process modeler 402 and BusinessWare server 404 are configured so that sub-process models may be labeled in the same way as business process models. Thus, there may be multiple versions of a sub-process model. Each definition may have an associated label including an "initial" and a "current" version. The checkpointing ability of Business process manager 408 is configured to include the version information for sub-process models during state saving operations. This allows business process objects to be restarted using the correct versions of their nested sub-process models.

Thus, Budhiraja does not provide the disclosure missing in Khodabakchian and Broussard as to claims 3, 11 and 25. This is an additional reason the rejection should be withdrawn against claims 3, 11 and 25.

It is respectfully submitted that the claimed subject matter is allowable over the cited references. Accordingly, Applicants respectfully request reconsideration and withdrawal of the rejection of claims 1-3 and 5-34 under 35 U.S.C. § 103(a) in view of various combinations of Khodabakchian, Broussard and Budhiraja.

CONCLUSION

Amendments, made herein or previously made, are without abandonment of subject matter. Applicant expressly reserves the right to, in the pending application or any application related thereto, reintroduce any subject matter removed from the scope of claims by any amendment and introduce any subject matter not present in current or previous claims.

In view of the foregoing remarks, it is respectfully submitted that this application is in condition for allowance. Reconsideration of this application and an early Notice of Allowance are respectfully requested.

Date: June 15, 2010

/Joseph F. Oriti/
Joseph F. Oriti
Registration No. 47,835

Woodcock Washburn LLP
Cira Centre
2929 Arch Street, 12th Floor
Philadelphia, PA 19104-2891
Telephone: (215) 568-3100
Facsimile: (215) 568-3439